Optimization of the Center-of-Mass Trajectory with Reinforcement Learning

Alexandros Paraschos & Thibaut Munzer

Problem Statement

- Transfer the CoM from left to right foot
- Analytical solution is hard to find
- Working Approximation
- Room for improvement

-> RL!

Trajectory represantation

- Dynamic Movement Primitives:

$$\gamma^2 \ddot{\boldsymbol{q}}_t = \alpha_q\big((\boldsymbol{g} - \boldsymbol{q}_t) - \beta_q \dot{\boldsymbol{q}}_t\big) + f_t$$

Spring-Damper System:

$\boldsymbol{g}$ . . . goal-attractor

α_q, β_q . . . spring and damping coefficients

Forcing function: $f_t = \boldsymbol{\phi}(z_t)^T \boldsymbol{w}$

$\boldsymbol{\phi}$ . . . Basis-Functions

$\boldsymbol{w}$ . . . Shape-Parameters

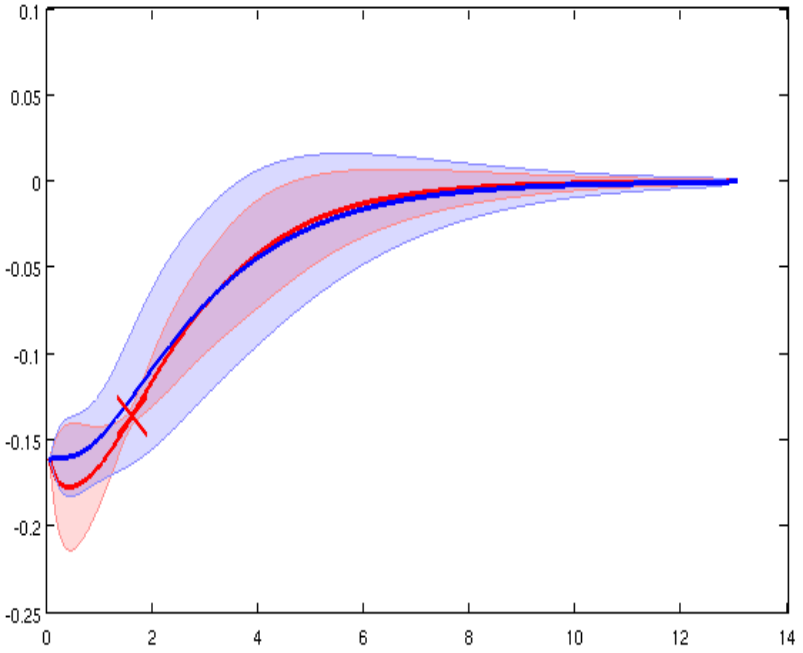
Phase variable: $\gamma \dot{z}_t = -\alpha z_t$

Replaces time, temporal scaling of the movement

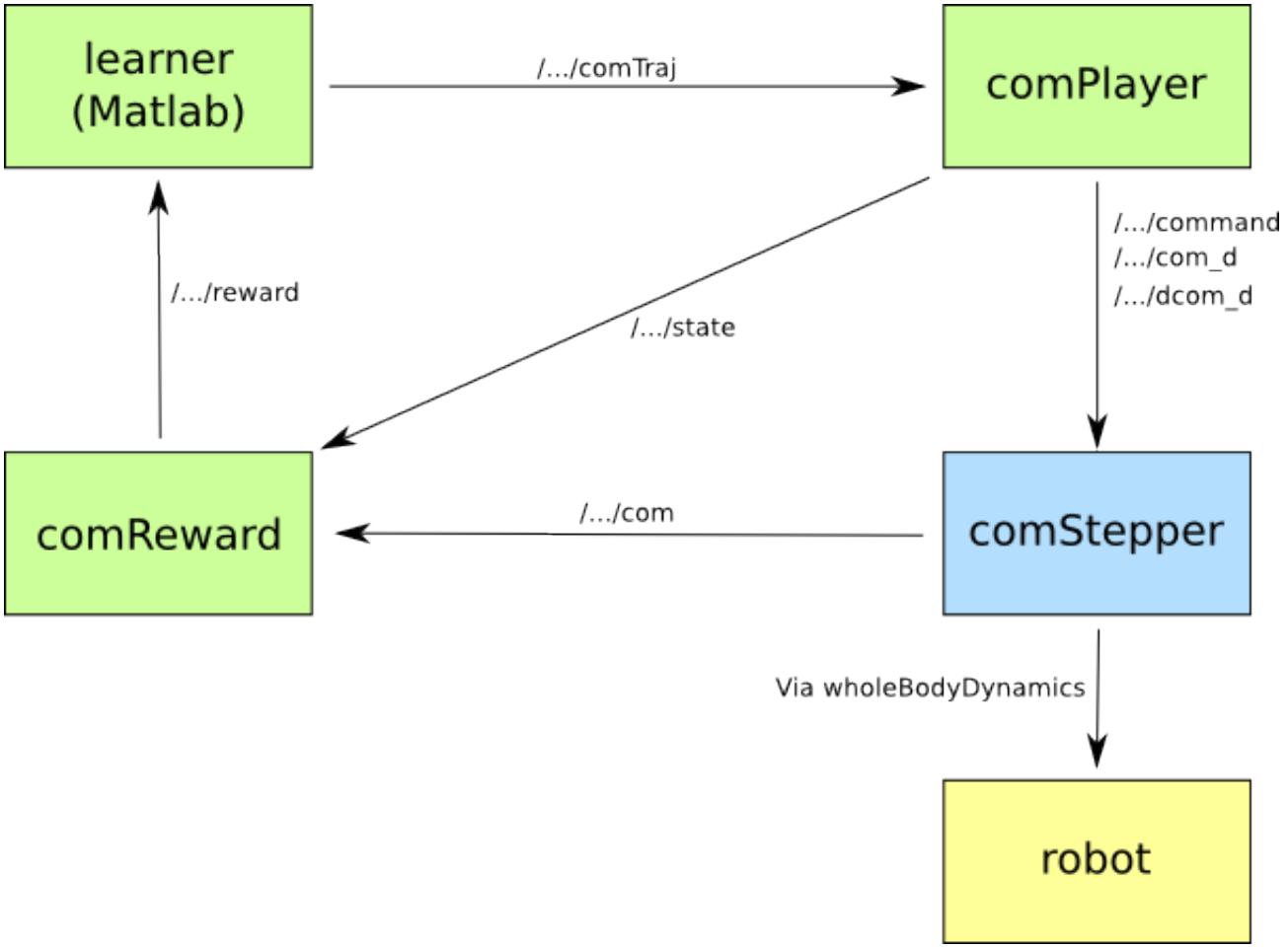
γ . . . temporal scaling factor

Reinforcement Learning Algorithm

- PoWER
  - Policy search
  - Maintain a distribution over the parameters
  - Samples new parameters
  - Weight each sample with the obtained reward
  - Update the distribution



RL Architecture



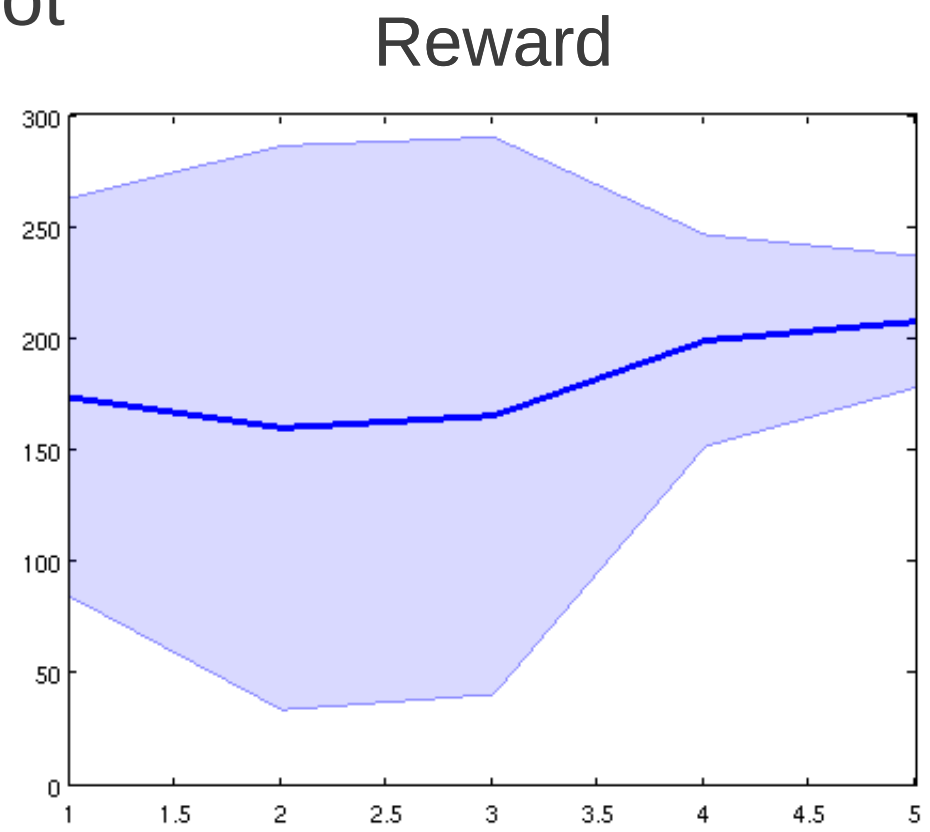
Experimental Setup

- Optimize the CoM projection (y, z)
  - 1 DMP for each dimension
  - 7 basis function per DMP
  - Plus the time coefficient
- PoWER algorithm
  - Episodic setup 10 samples per episode
- Initial policy: imitation of the analytical solution
- Reward
  - R = 750 – time not at objective * 0.02
  - Or R = 0 if the robot fall down during experiment

Results

- Not possible to run in simulation
- 5 Iterations on the real robot

	Reward
Demonstration	170
Demonstration (time scaled)	139
Best policy	231



Future work

- Evaluate for more iterations
- Average over multiple trials
- Learn to control the joint velocities
- Evaluate other RL algorithm
  - Bayesian optimisation, REPS, PI-BB
- Try different trajectory representations